# Apple Assembly Line

$1.50

Volume 2 -- Issue 10 July, 1982


In This Issue...


Run-Anywhere Subroutine Calls . . . . . 2
Cut the Bull Software . . . . . 3
Who Are We and What Are We Doing . . . . . 4
Giant Macro for Writing Messages . . . . . 6
Sorting Out Zero-Page References . . . . . 10
Axlon RAMDISK 320: A Review . . . . . 11
Simple Hi-Res Animation . . . . . 15
A Text File Display Command for DOS . . . . . 23
Hierographic Transport: A Review . . . . . 28
Christmas in July? . . . . . 32


## A New Software Tool: ES-CAPE

ES-CAPE stands for Extended S-C Applesoft Program Editor. You are somewhat familiar with it already as AED II from Linn Software. Bill has added more features, and I am in the process this month of re-doing the reference manual. I am shooting for it to be all packaged by the middle of July. The price will hold at $40 at least until September 1st.

If you are using Applesoft and feel the need for advanced editing tools to use on your programs-under-development, ES-CAPE ought to be in your tool-box. Like any tool, it doesn't do everything and it won't replace all your other tools. (You wouldn't try to tighten a screw with a hammer, or assemble a Heathkit with a SkilSaw....) But neither does it use all your money or memory!

## Current Advertising Rates

For the August 1982 issue the price will be $60 for a full page, $35 for a half page. To be included, I must receive your camera-ready copy by July 20th.

## Run-Anywhere Subroutine Calls.....Bob Sander-Cederlof

Bob Nacon (author of Amper-Magic) called yesterday and told me about his new way to call subroutines in programs that will be loaded anywhere in memory without relocation or reassembly. He does this a lot inside Amper-Magic, and you might want to do it yourself sometime.

Instead of JSR <subroutine name>, put the following three lines whenever you call a subroutine:

```
CLV
JSR $FF58
BVC <subroutine name>
```

The byte at $FF58 in the monitor ROM is always $60, an RTS instruction. Since this is used by most Apple interface boards, Apple has guaranteed that it will always be $60. The JSR to a guaranteed RTS instruction seems silly, doesn't it? Not quite, because it does put two bytes on the stack, and then pop them off again. But we can get them back later, inside the called subroutine, like this:

```
TSX    GET STACK POINTER
INX
DEX
TXS    REVISED STACK POINTER
```

Now the subroutine we called has a return address to go to, just as though we had used JSR <subroutine name>! The only problem is that if we execute an RTS, we will re-execute the BVC <subroutine name> and be in a loop. Unless....

Unless we set overflow, so the BVC falls through. But there is no SEV opcode in the 6502, so what do we do? $FF58 to the rescue again! Here is how we end the subroutine:

```
BIT $FF58    SET OVERFLOW
RTS
```

The BIT instruction copies bit 7 of $FF58 into the Carry Status bit, and bit 6 into the Overflow Status bit. This, in other words, (since $FF58 has $60 in it) clears carry and sets overflow. If you want carry to be set as a return flag, you can insert SEC between the BIT and RTS lines.

I thank Bob Nacon for this technique, and he thanks Roger Wagner for putting him on the trail to its discovery. Roger writes the monthly column in Softalk Magazine called "Assembly Lines"; the December, 1981, issue covered writing run-anywhere programs. If you haven't got Roger's book yet, called "Assembly Lines: The Book", it is currently the best book for beginners that I know of. The regular price is $19.95+$2 shipping, but I sell them for $18+$2 shipping.

Still More About "The Other Epson Reference Manual"

No sooner did I print my cutting comments about Cut The Bull Software last month, than I received a copy of the new edition of "The Other Epson Manual" in the mail. Bill Parker, author and publisher, has done an excellent job. By now all of you who ordered the booklet should have received your copy.

Bill has now quit his previous job to devote full time to the software company. The nature of that previous job prevented him from publishing his telephone number. Now you can reach him at (714) 223-3576. He says that in the future should a back order situation develop he will hold customer checks until ready to ship.

# *Cut The Bull Software*

***Box 82761 • San Diego, CA 92138***

Boy, the Bull really saw red after reading last month's Apple Assembly Line.

To clarify a few things, all back orders have been shipped, my home phone number is 223-3576, and we are now earning compliments on our speed and service.

We were swamped during the first two months of business. Seems we struck a nerve with the Other Epson Manual. Complicating things were our inexperience with starting a new business, the fact that the review didn't make clear what configuration of Epson the original Epson Manual was for, and the fact that there was just Mrs. Bull and me packing orders for 18 hours a day.

We now have The Epson Card, a series of flip charts for the MX-80 without Graftrax and The Other Epson Manual, a beautiful 40-page manual on the MX-80 with Graftrax 80. The manual includes many special features such as a HIRES screen dump, an address of a newsletter that shows you how to use your Epson with a variety of firmware and software, a reference card, etc. etc. More manuals are on the way.

To help say sorry for any inconvenience, we are offering MX-80 with Graftrax 80 Other Epson Manual owners an update to version 3.1 of the manual for only $5. (The manual costs $10). The disk will be updated for free; just send it and return postage. Or, if they prefer, customers can have a copy of the two most significant changes in the manual, the HIRES screen dump and the non-Epson interfacing & newsletter info for a stamped, self-addressed envelope and 25¢. Just include a note saying what you want.

Sincerely,

Bill Parker

Bill Parker, President
Cut The Bull Software

P.S. --Bill enclosed for 14 china stores destroyed in rampage.
-B.P.

Who are "we" and what are "we" doing?.....Mike Laumer

Some of you may wonder about the people whose articles you see in the AAL on a fairly regular basis and who you may have talked to on the phone at one time or another.

Bob Sander-Cederlof is the president of the S-C Software Corporation and the author of the S-C Assemblers and Double Precision Floating Point package. Bob has been working with computers since 1957, at such places as Control Data Corporation and Texas Instruments. He is developing a new text editor somewhat compatible with Apple Writer. Believe it or not the editor is half the size of Apple Writer. Both the editor and printer sections are in memory at once and it has more capabilities than Apple Writer. He also edits this newsletter every month, with the aid of Bill Morgan.

Bill Morgan is Bob's first full-time employee and helps in all areas: programming, shipping, accounting, phone sales, and writing articles for the AAL. He helps author the reference manuals as well, and tries to make our products fail before we start shipping them (so we can fix 'em before you see 'em!).

Bobby Deen is a part-time employee still in high school. He is currently helping Bob S-C develop a line of compatible Macro Cross Assemblers for 6800, 6809 and Z-80 processors to round out Bob's assembler product line. (The 6800 and 6809 versions are ready now.) He has helped develop an 18-digit decimal math package compatible with Applesoft soon to be a new product. He has also assisted in the CPR project with Mike Laumer.

Mike Laumer (that's me!) is owner of Laumer Research and author of FLASH! the Integer BASIC compiler, and of the upcoming MIKE'S MAGIC MATRIX hires graphics editor and animation design tool. As a sub-contractor to S-C Software for the last year, I have been working on an incredible application using Apples and video disks. You can read all about it in the June 1982 issue of BYTE magazine, pages 108-138. The American Heart Association sponsors the project, which will teach Cardiopulmonary Resuscitation (CPR). The Apple is supported by a video disk player, light pen, two CPR manikins, a random-access audio unit, and two monitors.

If you have called, you may have talked with Bob's daughter Patricia (oldest of five children). She is a Junior in High School, and works part-time at shipping, phone sales, mailing list maintenance, word processing, Visicalc-ing, program entry, paste-up and folding, and whatever comes up. She is assisted by Lisa MacCorkle, another high school friend.

We enjoy talking with all of you, so if you have a problem, need a book, or whatever, give us a call!

Giant Macro for Writing Messages.....Robert B. Steiner

Every time I turn around I seem to need a quick and dirty routine to print out a message. I must have written them a dozen different ways, to fill various requirements. Sometimes they are only different because of a silly mistake...a difference usually called a bug. I could keep a handful of them on a subroutine library, but then I might get mixed up as to which one was which.

S-C Macro Assembler to the rescue! By writing one of largest macros I have ever seen, I can get all the message-printer-variants into one neat little package. Then by choosing the correct parameters, the kind of printing routine I want is generated on the spot.

I call the macro CRT, and you call it with up to five parameters. The call line will look like one of these:

```
>CRT L,N,"your message"
>CRT L,I,"your message"
>CRT A,N,address of your message
>CRT A,I,address of your message
```

The first parameter, which may be "L" or "A", indicates whether you will give an actual message in quotation marks, or the address of the message.

The second parameter, which may be "N" or "I", stands for Normal or Inverse video display.

The third parameter is either the message itself in quotes, or the address of the message (a label, of course).

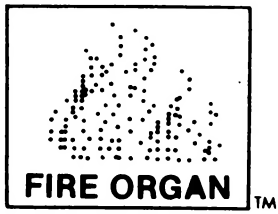
An optional fourth parameter may be "I", "Y", or "R". "I" will generate code to read an immediate one byte reply, which is returned in the A-register. "Y" will generate the one byte reply code, followed by additional code to check for a yes/no response. It will loop until you type "Y" or "N"; then it will echo the letter, print a RETURN, and return with the character in the A-register.

If the fourth parameter is "R",an entire line of reply is expected. If there is no fifth parameter, the line will be at $200 for your program to analyze. If a fifth parameter is used, it is the name of a buffer for the reply message.

If I counted correctly, there are twenty different possible ways the macro can be generated!

Here is the macro definition, and some sample call lines. Try it out; you'll find it fun and educational, whether its useful to you or not. Then you can apply some of the techniques in your own work.

```
1000 *--------------------------------
1010 * MACRO: >CRT SRC,DSPMODE,MSG
1020 *        >CRT SRC,DSPMODE,MSG,REPMODE
1030 * MACRO: >CRT SRC,DSPMODE,MSG,REPMODE,REPADDR
1040 *--------------------------------
1050         .MA CRT
1060         LDY #0          INITIALIZE INDEX
1070         .DO ']1='L      *** LITERAL MESSAGE ***
1080 :1      LDA :9,Y        GET MESSAGE CHARACTER
1090         .ELSE           *** ADDRESSED MESSAGE ***
1100 :1      LDA ]3,Y        GET MESSAGE CHARACTER
1110         .FIN
1120         PHA             SAVE CHARACTER ON STACK
1130 *--------------------------------
1140         .DO ']2='N      *** NORMAL DISPLAY ***
1150         ORA #$80        SET TOP BIT OF CHARACTER
1160         .ELSE           *** INVERSE DISPLAY ***
1170         AND #$3F
1180         .FIN
1190 *--------------------------------
1200         JSR $FDF0       DISPLAY CHARACTER
1210         INY             POINT TO NEXT CHARACTER
1220         PLA             GET ORIGINAL CHARACTER
1230         BMI :1          MORE IF TOP BIT = 1
1240 *--------------------------------
1250         .DO ']1='L      *** LITERAL ***
1260         BPL :2          ...ALWAYS
1270 :9      .AT -']3'       MESSAGE ITSELF
1280 :2
1290         .FIN
1300 *--------------------------------
1310         .DO ]#=3        *** DISPLAY ONLY ***
1320         LDA #$8D        CARRIAGE RETURN
1330         JSR $FDF0
1340         .ELSE
1350         .DO ']4='R      *** STRING REPLY EXPECTED ***
1360         LDA #$8D        CARRIAGE RETURN
1370         JSR $FDF0
1380         JSR $FD6F       READ REPLY
1390         .DO ]#=5        *** SPECIFY REPLY LOCATION ***
1400         LDY #0
1410 :3      LDA $200,Y      MOVE REPLY TO CALLER'S BUFFER
1420         STA ]5,Y
1430         INY
1440         CMP #$8D        WAS IT END OF LINE?
1450         BNE :3          MORE TO MOVE
1460         .FIN
1470         .ELSE
1480         LDA #$A0        ADD ONE BLANK TO MESSAGE
1490         JSR $FDF0
1500 :5      JSR $FD0C       GET REPLY CHARACTER
1510         .DO ']4='Y      *** Y/N REPLY ***
1520         CMP #'Y+$80
1530         BEQ :6
1540         CMP #'N+$80
1550         BNE :5          NEITHER Y NOR N
1560 :6
1570         .FIN
1580         PHA             SAVE REPLY
1590         JSR $FDF0       DISPLAY THE CHARACTER
1600         LDA #$8D
1610         JSR $FDF0       CARRIAGE RETURN
1620         PLA             GET REPLY CHARACTER
1630         .DO ']4='Y      *** Y/N REPLY ***
1640         CMP #'Y+$80     .EQ. IF "Y", .NE. IF "N"
1650         .FIN
1660         .FIN
1670         .FIN
1680         .EM
1690 *--------------------------------
1700         >CRT L,N,"ABCDEFG"
1710         >CRT L,I,"ABCDEFG"
1720         >CRT A,N,MSG
1730         >CRT A,I,MSG
1740 *--------------------------------
1750         >CRT L,N,"ABCDEFG",Y
1760         >CRT L,I,"ABCDEFG",I
1770         >CRT A,N,MSG,R
1780         >CRT A,I,MSG,R,BUFFER
1790         RTS
1800 MSG     .AT -/MESSAGE/
1810 BUFFER  .BS 256
```




-- APPLE OWNERS LOVED FIRE ORGAN (WHICH WAS FREE) --

"...AM FASCINATED BY FIRE ORGAN..."

STEVE WHEELER, AURORA CO

"THANK YOU FOR FIRE ORGAN. WE'VE HAD IT FOR 24 HOURS NOW AND HAVE NOT GOTTEN ANY WORK DONE."

DON MCMATH, TULSA OK

"..BEEN WATCHING FOR CEEMAC EVER SINCE I GOT THE FREE COPY OF FIRE ORGAN."

ED JACKSON, FORT WORTH TX

"..HAVE SHOWN FIRE ORGAN OVER LOCAL TV...(WITH)...MUSIC FROM A CABLE-ONLY, PROGRESSIVE RADIO STATION. ...IT WAS THE BEST SHOW WE'VE PUT ON YET."

JIM AIKIN, GAINESVILLE FL

"...HAVE SPENT MANY ENJOYABLE SUNDAY AFTERNOONS WITH FIRE ORGAN. THANKS MUCHLY."

B GARBEE, LYNCHBURG VA

"YOUR 'FIRE ORGAN' IS SO AMAZINGLY APPEALING..."

RODNEY A WHITE, GLENDALE CA

"...THINK THAT FIRE ORGAN IS ONE OF THE MOST IMPRESSIVE THINGS EVER DONE WITH AN APPLE."

AL EVANS, AUSTIN TX

"MY STUDENTS ARE DROOLING AT THE IDEA OF 'MAKING STUFF LIKE FIRE ORGAN' AND SO AM I."

MATT BERMAN, METAIRIE LA

"FIRE ORGAN IS TERRIFIC! WHERE AND HOW CAN I GET 'CEEMAC'?"

D WALKER, ALTAMONTE SPRINGS FL

-- NOW THEY'RE LOVING CEEMAC (WHICH ISN'T FREE) --
\-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --/

"A RECENT DEMONSTRATION OF (CEEMAC) ... REALLY EXCITED ME."

MICHAEL CALLERY, NEW YORK NY

"..YOU HAVE DEVELOPED THE SYSTEM I HAVE BEEN LOOKING FOR..."

BILL COZAD, KANSAS CITY MO

"FIRE ORGAN IS PROBABLY THE SINGLE MOST IMPRESSIVE USE OF THE APPLE GRAPHICS CAPABILITIES YET SEEN, AND WITH CEEMAC AVAILABLE SO THAT OTHERS CAN WORK WITH IT, THE PROGRAMMER'S CAPABILITIES WILL EXPAND A THOUSANDFOLD."

RICK E BERGER, PRINCETON NJ

"...YOU HAVE OBVIOUSLY TAKEN GRAPHICS TO A NEW STATE OF THE ART."

CHUCK CARLSON, TULSA OK

"THERE IS NO QUESTION IN MY MIND THAT CEEMAC IS THE MOST ORIGINAL, MOST USABLE APPLE GRAPHICS PACKAGE I HAVE SEEN (AND I THINK I HAVE SEEN THEM ALL)."

BILL DWYER, CUPERTINO CA

"YOU HAVE MADE A GREAT CONTRIBUTION TO BOTH ART AND SCIENCE WITH YOUR WORK."

M AINSWORTH-LAND, BUFFALO NY

"THANK YOU FOR THE TIME SPENT IN DEVELOPING A TRULY UNIQUE AND USEFUL PROGRAM."

BILL BOYDSTON, GLENDALE CA

"IT'S IMMEDIATELY OBVIOUS TO ME THAT CEEMAC IS EVERYTHING I HAD HOPED FOR.. AND MUCH MORE."

DOUG S YATSUHASHI, NORFOLK MA

SEE AND TRY CEEMAC AT YOUR NEARBY APPLE DEALER.

RELEASE 1.0 - $75

VAGABONDO ENTERPRISES
1300 E ALGONQUIN - 38
SCHAUMBURG IL - 60195

APPLE ][ IS A TRADEMARK OF APPLE COMPUTER, INC

CEEMAC AND FIRE ORGAN ARE TRADEMARKS OF VAGABONDO ENTERPRISES

## Sorting Out Zero-Page References.....Tracy L. Shafer

The search for page-zero references program in last month's AAL turned out to be (almost) the very thing I've been needing.

I have a clock card capable of generating NMI and IRQ interrupts. Up to now. I haven't been able to do any deep research on the IRQ due to the DOS and monitor conflict mentioned in the January issue of AAL. (They both use location $45.) I can't modify the monitor because I don't have access to a PROM burner, and the thought of searching through DOS really put a damper on the IRQ project until now.

Since I didn't need to know every page-zero reference used by DOS, I modified the program to search for a specific page-zero reference. That worked fine, but I didn't want to have to type in a separate search value for every group of references I might need later, so I further changed the program to print out all the references in numerical order of page-zero location.

To make the changes to the program as published last month, just remove the ".3" from line 1580 and add the following lines:

```
1285 PAGE.REF .HS 00           VARIABLE TO HOLD THE
                               CURRENT ZERO-PAGE
                               LOCATION

1571 .3       INY              NEW PLACE FOR ".3" LABEL
1572          LDA (MON.PCL),Y  GET PAGE REFERENCE
1573          DEY              RESTORE VALUE OF Y
1574          CMP PAGE.REF     ONE WE ARE SEARCHING FOR?
1575          BNE .6           NO, IGNORE THIS ONE

1861          LDX #1           RESTORE X-VALUE FOR
                               MON.A1PC ABOVE
1862          INC PAGE.REF     NEXT ZERO-PAGE ADDRESS
1863          BNE CTRL.Y       NOT FINISHED
```

The program now searches through the memory range 256 times instead of just once, so it doesn't run nearly as fast, but it's easier to find all the references to specific locations.

## AXLON RAMDISK 320: A Review.....Bill Morgan

AXLON's RAMDISK 320 is a system designed to add 320K of memory to an Apple, configured to look to the Apple like two very fast disk drives. The speed improvement ranges from half the time for a large assembly to one-twelfth the time for directly dumping 192 pages of memory.

### Hardware

The RAMDISK is a cabinet just the size of an Apple disk drive, containing the memory. its own power supply, and a backup battery. There is also a large interface card, which includes 2K of static RAM for the operating system.

The backup battery is said to provide up to three hours of protection against power outage. It did maintain power when we moved the system into another room (about 5 minutes), but you should certainly make floppy disk backups of the RAMDISK data before leaving the system unplugged overnight. As long as it is plugged into the wall, the battery is kept charged and the memory is maintained.

### Software

There are several programs supplied with the RAMDISK. These fall into the general categories of system software, utilities, and demonstrations.

RAMDSK1 is the operating system, which is stored in static RAM on the interface card, addressed in the $C800-CFFF space. BRUNning this program hooks it into DOS and copies one or two mechanical drives into the RAMDISK.

RDCOPY copies between the mechanical and RAM disks, to load or back up the RAMDISK. SELECT creates modified versions of RAMDSK1 for different slot/drive configurations.

The EXTRA40K utility allows you to access "tracks" 36-40 on the RAMDISK, but only on a level comparable to using RWTS directly. That is, you must work in terms of addresses and track/sectors rather than variables and filenames. The manual has a complete assembler source listing of this program.

SECTOR CHECKER and BYTE-BY-BYTE are self-test utilities to verify correct operation of the RAMDISK.

The demonstrations are The Directory and the Mini-Base Phone Book. The Directory is a large, disk-based, data-base program, in machine language, which uses the speed of the RAMDISK to its full advantage. The problem with this program is that it is strictly fixed-format, with no provision for modifying the record structure. The fields built into a record are last name, first name, dept #, mail stop, phone, special interest 1, special interest 2, and comments. If you are a large company needing an on-line, internal phone directory, then The

Directory is outstanding. Otherwise, it's just an interesting demonstration of the system's capabilities.

The Mini-Base Phone Book is a memory-based data base, somewhat similar to File Cabinet. The Mini-Base is also set up as an internal phone directory. but since it is written in Applesoft, it can be modified to suit your needs. The documentation includes instructions for changing the record structure. The manual also contains instructions for calling special machine-language routines for keyboard input, fast loading of text files (in a specified format), and fast sorting of a string array.

## Documentation

The manual is in three sections: 63 pages on the system, 34 pages on The Directory program, and 43 pages on the Mini-Base Phone Book program. It all comes in a large (8 1/2 by 11) 3-ring binder. The system section has chapters on setting up the RAMDISK, using the included software, calling it from DOS 3.3, attaching and using it in Pascal, technical information, and accessing the system from assembly language.

The setup and software chapters are quite good; the DOS chapter just says that everything is standard. I don't have Pascal, so I can't evaluate that section. The technical and assembly language chapters have all the information about memory usage, addressing, and programming techniques needed to use the RAMDISK without all of DOS's overhead.

## Using the RAMDISK

To use the RAMDISK with your programs, you need to copy the RAMDSK1 program onto your disk and set up the HELLO program to BRUN RAMDSK1. This will load the operating system into the interface card, then fast-copy your disk into drive one of the RAMDISK. Once your information is loaded into the RAMDISK, you can use all the normal DOS techniques to read and write files; the only difference is speed.

You can avoid the DOS overhead either by calling RWTS in the usual manner, or by directly using the RAMDISK registers and memory window. To do that, you just store track, sector, and drive information into two bytes, then read the data from $C800-C8FF. This approach is fastest. but you must then take on all memory management chores. Appendices to the manual list assembler source code for routines using both techniques.

## The Negative Side

We discovered one apparent bug in the RAMDISK's operating system. The program does not properly update the slot and drive found parameters in the I/O Control Block used by RWTS. If a program tries to use those locations to determine which drive it was run from. it will get the wrong data.

Mechanical disk drives are known to be error-prone, so DOS has some built-in protection against errors. Each sector is recorded with a checksum; when a sector is read the checksum should match. This is very poor protection, but it does catch most errors. The RAMDISK has no such protection. The RAMDISK is much less likely to have any errors than the mechanical drives, yet it still would be nice to have at least a sector checksum. Parity on each byte would be even better, but it would be expensive.

Timing Comparisons

| Operation | Disk II time | RAMDISK time |
|---|---|---|
| Assemble 102 sectors of source code. | 89 sec. | 41 sec. |
| BLOAD Hi-res screen. | 11 sec. | 3 sec. |
| LOAD Applesoft program. | 14 sec. | 4 sec. |
| Dump RAM (192 sectors) calling RWTS. | 9 sec. | .8 sec. |
| Dump 192 sectors direct | n/a | .7 sec. |

Summary

The RAMDISK is a well-made and well-documented unit; it performs as advertised. The RAMDISK gives a terrific speed improvement over mechanical disk drives, especially if you do your own reading and writing and avoid DOS.

Two standard Apple drives with controller at normal retail prices would cost $1180; RAMDISK goes for $1395, and you get the equivalent of 10 extra tracks thrown in. (On the other hand, several non-Apple drives are available with 40 to 80 tracks, at competitive prices. And the 5- and 10-megabyte Winchesters are rapidly falling in price.)

I have seen RAMDISK advertised for as low as $1170 in Byte Magazine.

The RAMDISK 320 is available from AXLON, Inc., 170 N. Wolfe Rd., Sunnyvale, CA 94086, (408) 730-0216. RAMDISK 320, The Directory. and Mini-Base Phone Book are trademarks of AXLON INC.

# QUICKTRACE

**relocatable program traces and displays the actual machine operations, *while* it is running without interfering with those operations. Look at these *FEATURES:***

***Single-Step** mode displays the last instruction, next instruction, registers, flags, stack contents, and six user-definable memory locations.*

***Trace** mode gives a running display of the Single-Step information and can be made to stop upon encountering any of nine user-definable conditions.*

***Background** mode permits tracing with no display until it is desired. Debugged routines run at near normal speed until one of the stopping conditions is met, which causes the program to return to Single-Step.*

***QUICKTRACE** allows changes to the stack, registers, stopping conditions, addresses to be displayed, and output destinations for all this information. All this can be done in Single-Step mode while running.*

***Two optional display formats** can show a sequence of operations at once. Usually, the information is given in four lines at the bottom of the screen.*

***QUICKTRACE** is completely transparent to the program being traced. It will not interfere with the stack, program, or I/O.*

***QUICKTRACE** is relocatable to any free part of memory. Its output can be sent to any slot or to the screen.*

***QUICKTRACE** is completely compatible with programs using Applesoft and Integer BASICs, graphics, and DOS. (Time dependent DOS operations can be bypassed.) It will display the graphics on the screen while **QUICKTRACE** is alive.*

***QUICKTRACE** is a beautiful way to show the incredibly complex sequence of operations that a computer goes through in executing a program*

**QUICKTRACE** *$50*

Is a trademark of Anthro-Digital, Inc.



*See these programs at participating Computerland and other fine computer stores.*

**Anthro - Digital Software, Inc.**
**P.O. Box 1385 Pittsfield, MA 01202**

Simple Hires Animation.....Mike Laumer

One thing that I have been working with in my next product (MIKE'S MAGIC MATRIX) is animation using hires graphics. I have been developing a hires graphics editor using the FLASH! Integer BASIC Compiler. I may not be the first one to bring a commercial product to market using the FLASH! compiler since there are at least six other programmers who are striving to beat me.

There are several methods used to achieve animation in the popular game programs. The one presented in this program is possibly the simplest. This program will animate an image in one place on the screen (in-place animation) from a series of frames of data.

The technique used to display the frame data on the screen is simply moving the data with 'LDA' and 'STA' instructions. A more powerful method of animation is to use the 'EOR' instruction to merge one frame of animation into the next. This is accomplished by using the frame data obtained by 'EOR'ing two successive frames of data. Then using that new data to 'EOR' to the image data. The 'EOR' istruction is very useful since it can add and delete data to and from the screen without disturbing any background that may be on the screen already.

A frame of data for the animation is written to the screen and then a delay loop entered to delay before the next data frame is written to the screen. If the delay is smaller the animation will speed up. If the delay is larger the animation will slow down. The delay could be read from the game paddle.

The method I used in the routine to compute the hires graphics screen addresses is to use two tables (one for lo-byte, one for hi-byte) with 192 entries to convert the Y-coordinate into a hires address. Otherwise, the Y-addresses would have to be computed by using a complicated formula:

```
A = Y MOD 8
B = (Y / 8) MOD 8
C = Y / 64
YADRS = 8192 + A*1024 + B*128 + C*40
(add another 8192 if hires page2 adress needed)
```

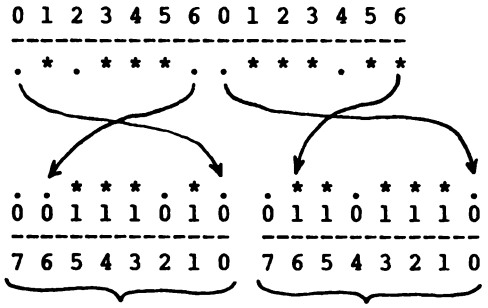
So you see that even with an efficiently coded machine language routine to compute a screen address it will take a bit of time to perform. It is much more effecient to simply look up the address of the first byte of the Y-row in a table. Since the Y-coordinate never exceeds 191 (which is less than 256) the Y-register can be used easily to index the table. The table in the program only provides the offset from the beginning of a hires page. The program uses an 'ORA' instruction to put $20 or $40 into the hi-byte to specify hires page 1 or 2.

The data for the animations were built with MIKE'S MAGIC MATRIX and the first frame looks like this:

| | | | | | |
|---|---|---|---|---|---|
| ..... | 00000000 | 0 | 0 | 0 | 0 |
| ..... | 00000000 | 0 | 0 | 0 | 0 |
| .....**** | 00600300 | 0 | 96 | 3 | 0 |
| .....****** | 00700700 | 0 | 112 | 7 | 0 |
| .....**.**.** | 00580D00 | 0 | 88 | 13 | 0 |
| .....******** | 00780F00 | 0 | 120 | 15 | 0 |
| .....***..*** | 00380E00 | 0 | 56 | 14 | 0 |
| .....****** | 00700700 | 0 | 112 | 7 | 0 |
| .....**** | 00600300 | 0 | 96 | 3 | 0 |
| .....** | 00400100 | 0 | 64 | 1 | 0 |
| .....** | 00400100 | 0 | 64 | 1 | 0 |
| .....******** | 00780F00 | 0 | 120 | 15 | 0 |
| .....********** | 007C1F00 | 0 | 124 | 31 | 0 |
| .....**..****..** | 00663300 | 0 | 102 | 51 | 0 |
| .....**....**....** | 00436100 | 0 | 67 | 97 | 0 |
| .....**...****...** | 00636300 | 0 | 99 | 99 | 0 |
| .....**..******..** | 00736700 | 0 | 115 | 103 | 0 |
| .....**...******...** | 40714701 | 64 | 113 | 71 | 1 |
| .....**..***..***..** | 40394E01 | 64 | 57 | 78 | 1 |
| .....**....** | 00180C00 | 0 | 24 | 12 | 0 |
| .....**....** | 00180C00 | 0 | 24 | 12 | 0 |
| .....**....** | 00180C00 | 0 | 24 | 12 | 0 |
| .....**....** | 00180C00 | 0 | 24 | 12 | 0 |
| .....**....** | 00180C00 | 0 | 24 | 12 | 0 |

The data was written to a text file from within the editor and run through an Applesoft program to create an EXEC file for the S-C Macro Assembler to insert the data tables into the program.

You can make your own frames of animation by a hand process of drawing the animation dots on graph paper and reducing the data into hexadecimal data. To do this you must take each row of dots (on or off) on the graph paper and take them 7 dots at a time. The 7 dots must then be flipped into reverse order before converting into hex. Here is an example of 14 pixels width:




As you can see the process is a pain in the neck. If the animation has a flaw in it you have to repeat the process for every frame of data that is wrong. That is where a hires graphics editor and animation design tool like MIKE'S MAGIC MATRIX really shines, because you can perfect your animation and test it in the editor without ever leaving. MIKE'S MAGIC MATRIX is not yet ready for sale lacking a manual and a little more work. I expect to have the first version ready in about two more months. Preliminary showings to the Dallas Apple Corps indicated an enormous popularity.

Since hexadecimal strings take up a lot of listing space when they are assembled, I decided to print the tables here using just the LIST command, without the assembled object code listing. The program part is shown in the normal assembled format.

Here is what you will see if you get it all typed correctly:

Of course, they will all appear one after the other in the same screen position, not side-by-side.

```
                        1000 *--------------------------------
                        1010 *--------------------------------
                        1020 * SIMPLE ANIMATION
                        1030 *--------------------------------
FCA8-                   1040 MON.WAIT     .EQ $FCA8     MONITOR DELAY ROUTINE
                        1050 *
0000-                   1060 T1           .EQ $0,1
0002-                   1070 T2           .EQ $2,3
0004-                   1080 T3           .EQ $4,5
0006-                   1090 Y.INDEX      .EQ $6,7
                        1100 *--------------------------------
                        1110 * ANIMATION PLAYBACK LOCATIONS
                        1120 *--------------------------------
0020-                   1130 HIRES.PAGE   .EQ $20       $20 = PAGE 1, $40 = PAGE 2
0064-                   1140 Y.COORD      .EQ 100       WHERE TO PUT ANIMATION
0014-                   1150 X.COORD      .EQ 20        WHERE TO PUT ANIMATION
                        1160 *--------------------------------
                        1170         .OR $803
                        1180         .TF B.ANIMATE
                        1190 *--------------------------------
0803- 20 3C 08          1200 START   JSR HIRES.INIT     INITIALIZE HIRES SCREEN
0806- 20 0C 08          1210 .1      JSR PLAY.FRAMES    PLAY 1 SET OF FRAMES
0809- 4C 06 08          1220         JMP .1             GO DO IT AGAIN
                        1230 *--------------------------------
080C- A9 00             1240 PLAY.FRAMES LDA #0         INIT FRAME INDEX
080E- 8D 74 08          1250         STA FRAME.INDEX
0811- AD 74 08          1260 .1      LDA FRAME.INDEX    GET FRAME INDEX POINTER
0814- C9 09             1270         CMP #NUM.FRAMES    HAVE ALL FRMES BEEN DONE
0816- F0 23             1280         BEQ .3             YES, SO RETURN
0818- AC 00 C0          1290         LDY $C000      HAS A KEY BEEN PRESSED
081B- 10 09             1300         BPL .2         NO, SO KEY PLAYING THE FRAMES
081D- AD 51 C0          1310         LDA $C051      RESTORE TEXT SCREEN
0820- AD 54 C0          1320         LDA $C054      PRIMARY PAGE
0823- 4C D0 03          1330         JMP $3D0       EXIT ON ANY KEY
0826- 0A                1340 .2      ASL            DOUBLE INDEX
0827- A8                1350         TAY
0828- B9 62 08          1360         LDA FRAME.TABLE,Y  GET TABLE ADDRESS
082B- 85 00             1370         STA T1         SAVE ADRS IN T1
082D- C8                1380         INY            NEXT BYTE OF ADRS
082E- B9 62 08          1390         LDA FRAME.TABLE,Y
0831- 85 01             1400         STA T1+1
0833- 20 75 08          1410         JSR ANIMATE    MOVE FRAME DATA TO SCREEN
0836- EE 74 08          1420         INC FRAME.INDEX NEXT FRAME
0839- D0 D6             1430         BNE .1         ...ALWAYS
083B- 60                1440 .3      RTS
                        1450 *--------------------------------
083C- A9 20             1460 HIRES.INIT LDA #HIRES.PAGE
083E- 85 01             1470         STA T1+1
0840- A0 00             1480         LDY #0
0842- 84 00             1490         STY T1
0844- 98                1500 .0      TYA            ZERO A REG
0845- 91 00             1510 .1      STA (T1),Y     CLEAR SCREEN PAGE
0847- C8                1520         INY
0848- D0 FB             1530         BNE .1
084A- E6 01             1540         INC T1+1       NEXT PAGE
084C- A5 01             1550         LDA T1+1       CHECK FOR
084E- 29 1F             1560         AND #$1F       END OF HIRES PAGE
0850- D0 F2             1570         BNE .0         NO, CLEAR MORE
0852- AD 50 C0          1580         LDA $C050      ENABLE GRAPHICS
0855- AD 57 C0          1590         LDA $C057      ENABLE HIRES
0858- AD 54 C0          1600         LDA $C054      ENABLE PAGE 1 (C055 IS PAGE 2)
085B- AD 52 C0          1610         LDA $C052      NOMIX
085E- 60                1620         RTS
                        1630 *--------------------------------
085F- 14                1640 INTER.FRAME.DELAY    .DA #20
0860- 04                1650 XSIZE   .DA #4         X FRAME SIZE IN BYTES
0861- 18                1660 YSIZE   .DA #24        Y FRAME SIZE IN BYTES
                        1670 FRAME.TABLE
0862- 3F 0A             1680         .DA FRAME1
0864- 9F 0A             1690         .DA FRAME2
0866- FF 0A             1700         .DA FRAME3
0868- 5F 0B             1710         .DA FRAME4
086A- BF 0B             1720         .DA FRAME5
086C- 1F 0C             1730         .DA FRAME6
086E- 7F 0C             1740         .DA FRAME7
0870- DF 0C             1750         .DA FRAME8
0872- 3F 0D             1760         .DA FRAME9
0009-                   1770 NUM.FRAMES .EQ 9
0874- 00                1780 FRAME.INDEX .DA #0
```

```
                     1790 *--------------------------------
0875- A9 64          1800 ANIMATE LDA #Y.COORD      THIS IS THE STARTING ROW
0877- 85 06          1810         STA Y.INDEX       FOR THE ANIMATION
0879- AC 61 08       1820         LDY YSIZE     NUMBER OF ROWS TO PUT ON SCREEN
087C- 84 02          1830         STY T2
087E- A4 06          1840 .1      LDY Y.INDEX
0880- B9 BF 08       1850         LDA YTBL.LO,Y     COMPUTE THE ROW ADRS
0883- 18             1860         CLC
0884- 69 14          1870         ADC #X.COORD      ADD THE X OFFSET
0886- 85 04          1880         STA T3
0888- B9 7F 09       1890         LDA YTBL.HI,Y
088B- 69 20          1900         ADC #HIRES.PAGE   ADD THE HIRES PAGE BITS
088D- 85 05          1910         STA T3+1          T3 POINTS TO ROW POSITION
088F- AC 60 08       1920         LDY XSIZE         NUMBER OF BYTES TO PUT INTO ROW
0892- 88             1930         DEY               INDEX BEGINS AT ZERO TO XSIZE-1
0893- B1 00          1940 .3      LDA (T1),Y        GET FRAME DATA
0895- 91 04          1950         STA (T3),Y        PUT ONTO SCREEN
0897- 88             1960         DEY               FOR ALL BYTES IN THE ROW
0898- 10 F9          1970         BPL .3
089A- E6 06          1980 .4      INC Y.INDEX       NEXT ROW INDEX
089C- A5 00          1990         LDA T1
089E- 18             2000         CLC
089F- 6D 60 08       2010         ADC XSIZE         STEP FRAME ADRS AHEAD
08A2- 85 00          2020         STA T1            TO NEXT ROW OF DATA
08A4- A5 01          2030         LDA T1+1
08A6- 69 00          2040         ADC #0
08A8- 85 01          2050         STA T1+1
08AA- C6 02          2060         DEC T2            COUNT DOW THE ROWS
08AC- D0 D0          2070         BNE .1            GO MOVE REST OF FRAME ROWS
08AE- AC 5F 08       2080         LDY INTER.FRAME.DELAY
08B1- F0 0B          2090         BEQ .6            NO DELAY BETWEEN FRAMES
08B3- 84 02          2100         STY T2            SAVE DELAY
08B5- A9 1E          2110 .5      LDA #30           REPEAT THIS SMALL DELAY
08B7- 20 A8 FC       2120         JSR MON.WAIT
08BA- C6 02          2130         DEC T2            FOR COUNT IN 'T2'
08BC- D0 F7          2140         BNE .5            MORE DELAY
08BE- 60             2150 .6      RTS               FRAME IS ALL DONE
```

```
2160 *-------------------------------
2170 * HIRES Y OFFSET TABLES
2180 * OFFSET FROM $2000 OR $4000
2190 * HIRES PAGE DISPLAYS
2200 * USING THESE TABLES SPEEDS UP
2210 * HIRES SCREEN ADRS COMPUTATION
2220 * A GREAT DEAL!
2230 *
2240 * FOR EVERY Y VALUE FROM 0-191
2250 * THERE IS AN ENTRY IN THIS TABLE
2260 * TO COMPUTE THE ADRS OF FIRST
2270 * BYTE IN THE ROW.
2280 *-------------------------------
2290 YTBL.LO   .EQ *
2300   .HS 0000000000000000808080808080808080
2310   .HS 0000000000000000808080808080808080
2320   .HS 0000000000000000808080808080808080
2330   .HS 0000000000000000808080808080808080
2340   .HS 2828282828282828A8A8A8A8A8A8A8A8A8
2350   .HS 2828282828282828A8A8A8A8A8A8A8A8A8
2360   .HS 2828282828282828A8A8A8A8A8A8A8A8A8
2370   .HS 2828282828282828A8A8A8A8A8A8A8A8A8
2380   .HS 5050505050505050D0D0D0D0D0D0D0D0D0
2390   .HS 5050505050505050D0D0D0D0D0D0D0D0D0
2400   .HS 5050505050505050D0D0D0D0D0D0D0D0D0
2410   .HS 5050505050505050D0D0D0D0D0D0D0D0D0
2420 YTBL.HI   .EQ *
2430   .HS 0004080C1014181C0004080C1014181C
2440   .HS 0105090D1115191D0105090D1115191D
2450   .HS 02060A0E12161A1E02060A0E12161A1E
2460   .HS 03070B0F13171B1F03070B0F13171B1F
2470   .HS 0004080C1014181C0004080C1014181C
2480   .HS 0105090D1115191D0105090D1115191D
2490   .HS 02060A0E12161A1E02060A0E12161A1E
2500   .HS 03070B0F13171B1F03070B0F13171B1F
2510   .HS 0004080C1014181C0004080C1014181C
2520   .HS 0105090D1115191D0105090D1115191D
2530   .HS 02060A0E12161A1E02060A0E12161A1E
2540   .HS 03070B0F13171B1F03070B0F13171B1F
2550 *-------------------------------
2560 * ANIMATION DATA
2570 *-------------------------------
2580 FRAME1
2590   .HS 000000000000000000600300
2600   .HS 0070070000580D0000780F00
2610   .HS 00380E0000700700000600300
2620   .HS 0040010000400100000780F00
2630   .HS 007C1F0000663300000436100
2640   .HS 0063630000736700040714701
2650   .HS 40304E0100180C0000180C00
2660   .HS 00180C0000180C0000180C00
2670 FRAME2
2680   .HS 000000000060030000700700
2690   .HS 00580D0000780F0000380E00
2700   .HS 0070070000600300000400100
2710   .HS 0040010000780F00007C1F00
2720   .HS 0066330040414101 6C60031B
2730   .HS 3C70071E0070070000380E00
2740   .HS 00180C00000C1800000C1800
2750   .HS 000C1800000C180000000000
2760 FRAME3
2770   .HS 0060030000700700000580D00
2780   .HS 00780F0000380E0000700700
2790   .HS 0660033006400130 0C400118
2800   .HS 78780F0E607F7F0740677301
2810   .HS 0040010000600300000700700
2820   .HS 0070070001C1C00000C1800
2830   .HS 000C1800000C180000063000
2840   .HS 0006300000000000000000000
2850 FRAME4
2860   .HS 000000004061430140714701
2870   .HS 60580D0320780F0220380E02
2880   .HS 6070070340614301004 14100
2890   .HS 00463100007E3F0000780F00
2900   .HS 0060030000400100000600300
2910   .HS 0070070000700700001C1C00
2920   .HS 0006300000036000000036000
2930   .HS 4001400140014001000000000
2940 FRAME5
2950   .HS 0000000000000000406 10303
2960   .HS 4071070320580D0220780F02
2970   .HS 60380E0360700703406 14301
2980   .HS 0043610000463100007E3F00
2990   .HS 00780F0000600300004 00100
3000   .HS 0060030000700700000700700
3010   .HS 001C1C0000063000000036000
3020   .HS 0003600040014001400 14001
3030 FRAME6
3040   .HS 000000004061430140714701
3050   .HS 60580D0320780F0260380E03
3060   .HS 6070070340614301004 36100
3070   .HS 00463100007E3F0000780F00
3080   .HS 0060030000400100006 00300
3090   .HS 0070070000700700001C1C00
3100   .HS 0006300000036000000036000
3110   .HS 4001400140014001000000000
3120 FRAME7
3130   .HS 406143014071470160580D03
3140   .HS 20780F0220380E0260700703
3150   .HS 4061430100436100004 63100
3160   .HS 007E3F0000780F0000600300
3170   .HS 0040010000600300007 00700
3180   .HS 00700700001C1C0000063000
3190   .HS 0003600000036000400 14001
3200   .HS 4001400100000000000000000
3210 FRAME8
3220   .HS 0060030000700700005 80D00
3230   .HS 00780F0000380E0000700700
3240   .HS 0660033006400130 0C400118
3250   .HS 78780F0E607F7F0740677301
3260   .HS 0040010000600300007 00700
3270   .HS 00700700001C1C00000C1800
3280   .HS 000C1800000C180000063000
3290   .HS 0006300000000000000000000
3300 FRAME9
3310   .HS 000000000060030000700700
3320   .HS 00580D0000780F0000380E00
3330   .HS 0070070000600300004 00100
3340   .HS 0040010000780F00007C1F00
3350   .HS 0066330040414101 6C60031B
3360   .HS 3C70071E0070070000380E00
3370   .HS 00180C00000C1800000C1800
3380   .HS 000C1800000C180000000000
```

## A Text File Display Command for DOS.....Bob Sander-Cederlof

How many times have you wished that you could see what was in a TEXT file? You end up loading a word processor (if you are lucky enough to have one that can read normal DOS TEXT files), or EXECing it into the S-C Macro Assembler, or writing a special Applesoft program.... Why not a DOS command for this very common need?

The June 1982 issue of Call A.P.P.L.E. has an article by Lee Reynolds describing the addition of a FILEDUMP command to DOS. Lee gives a 20-byte program which fits nicely in an unused space in DOS. He replaced the MAXFILES command with "FILEDUMP". In case you want to try it, here are the patches for Lee's method.

```
]CALL -151
*BCDF:20
*BCE0:8E FD 20 A3 A2 20 8C A6 F0 05 20 F0 FD D0 F6 20
*BCF0:FC A2 60
*A8E7:46 49 4C 45 44 55 4D D0
*9D48:DE BC
*A933:20 30
*3D0G
]
```

$BCDF-BCF2 is the FILEDUMP command processor. $A8E7-$A8EE is the string "FILEDUMP", the command name. The two bytes at $9D48,9D49 are the address (minus 1) of the command processor. The two bytes at $A933,A934 are flags indicating that the FILEDUMP command requires a filename, and can optionally have S and D parameters.

My first reaction to the program, being a programmer, was to try to modify it. The first change I made saved one byte. The last two instructions are a JSR and an RTS. By ending with a JMP to the final subroutine, the RTS at BCF2 is not needed. Then I tried modifying the order of the loop, and saved another three bytes. Here is my revised listing:

```
                    1000 *--------------------------------
                    1010 *       "FILEDUMP" COMMAND
                    1020 *--------------------------------
A2A3-               1030 DOS.OPEN.TEXT.FILE   .EQ $A2A3
A2FC-               1040 DOS.CLOSE.FILE       .EQ $A2FC
A68C-               1050 DOS.READ.ONE.BYTE    .EQ $A68C
FDF0-               1060 MON.COUT1            .EQ $FDF0
                    1070 *--------------------------------
                    1080         .OR $BCDF
                    1090         .TA $8DF
                    1100 FILEDUMP
BCDF- 20 A3 A2      1110         JSR DOS.OPEN.TEXT.FILE
BCE2- A9 8D         1120         LDA #$8D
BCE4- 20 F0 FD      1130 .1      JSR MON.COUT1
BCE7- 20 8C A6      1140         JSR DOS.READ.ONE.BYTE
BCEA- D0 F8         1150         BNE .1       PRINT IT
BCEC- 4C FC A2      1160         JMP DOS.CLOSE.FILE
                    1170 *--------------------------------
                    1180         .OR $A8E7
                    1190         .TA $8E7
A8E7- 46 49 4C
A8EA- 45 44 55
A8ED- 4D D0         1200         .AT /FILEDUMP/    NAME OF FILEDUMP COMMAND
                    1210 *--------------------------------
                    1220         .OR $9D48
                    1230         .TA $848
9D48- DE BC         1240         .DA FILEDUMP-1    BRANCH FOR FILEDUMP COMMAND
                    1250 *--------------------------------
                    1260         .OR $A933
                    1270         .TA $833
A933- 20 30         1280         .HS 2030      FILENAME REQUIRED, SLOT & DRIVE
                    1290 *                     ARE OPTIONAL
```

After playing with the new command a little, I thought of several more changes. I wanted to be able to stop the file listing, to restart it, and to abort it. The first article I ever wrote about Apples described just such an addition, at that time for Integer BASIC. (See MICRO, June/July, 1978.) With this addition, the program would not fit in the unused space at $BCDF, so I decided to put it in the place of the INIT command instead. I changed the name to "SHOW".

Not all of the code would fit in the spot where the INIT command processor is, at $A54F. Therefore I broke out the routine to check for the pause/abort keys as a separate subroutine, and placed in over the top of some of the INIT code inside the File Manager of DOS. If you install this patch, you could call on the PAUSE.CHECK subroutine from your own programs.

```
                    1000 *--------------------------------
                    1010 *       "SHOW" COMMAND
                    1020 *--------------------------------
A2A3-               1030 DOS.OPEN.TEXT.FILE  .EQ $A2A3
A2FC-               1040 DOS.CLOSE.FILE      .EQ $A2FC
A68C-               1050 DOS.READ.ONE.BYTE   .EQ $A68C
C000-               1060 KEYBOARD            .EQ $C000
C010-               1070 STROBE              .EQ $C010
FDF0-               1080 MON.COUT1           .EQ $FDF0
                    1090 *--------------------------------
                    1100         .OR $A54F
                    1110         .TA $84F
                    1120 SHOW
A54F- 20 A3 A2      1130         JSR DOS.OPEN.TEXT.FILE
A552- A9 8D         1140         LDA #$8D
A554- 20 F0 FD      1150 .1      JSR MON.COUT1
A557- 20 8E AE      1160         JSR PAUSE.CHECK
A55A- F0 05         1170         BEQ .2
A55C- 20 8C A6      1180         JSR DOS.READ.ONE.BYTE
A55F- D0 F3         1190         BNE .1          PRINT IT
A561- 4C FC A2      1200 .2      JMP DOS.CLOSE.FILE
                    1210 *--------------------------------
                    1220 *       RETURN .EQ. IF ABORT
                    1230 *              .NE. IF CONTINUE
                    1240 *--------------------------------
                    1250         .OR $AE8E           OVER "INIT" CODE
                    1260         .TA $88E
                    1270 PAUSE.CHECK
AE8E- AD 00 C0      1280         LDA KEYBOARD        ANY KEY PRESSED?
AE91- 10 11         1290         BPL .2              NO, CONTINUE
AE93- 8D 10 C0      1300         STA STROBE          YES, CLEAR STROBE
AE96- C9 8D         1310         CMP #$8D            ABORT?
AE98- F0 0A         1320         BEQ .2              YES, RETURN .EQ. STATUS
AE9A- AD 00 C0      1330 .1      LDA KEYBOARD        NO, PAUSE TILL KEYPRESS
AE9D- 10 FB         1340         BPL .1              NONE PRESSED YET
AE9F- 8D 10 C0      1350         STA STROBE          CLEAR STROBE
AEA2- C9 8D         1360         CMP #$8D            ABORT?
AEA4- 60            1370 .2      RTS                 .EQ. IF ABORT
                    1380 *--------------------------------
                    1390         .OR $A884
                    1400         .TA $884
A884- 53 48 4F
A887- D7            1410         .AT /SHOW/    SHOW COMMAND NAME
                    1420 *--------------------------------
                    1430         .OR $A909
                    1440         .TA $809
A909- 20 30         1450         .HS 2030      FLAGS FOR SHOW COMMAND
                    1460 *--------------------------------
```

After assembling the program above, the various pieces are in memory in page 8 and 9, instead of inside DOS. I did it this way because DOS is protected during assembly. You can install the patches by hex input commands, or by some memory moves. I did it this way:

```
:$A54F<84F.863M
:$AE8E<88E.8A4M
:$A884:53 48 4F D7
:$A909:20 30
```

Then try typing "SHOW filename", where "filename" is a text file, and see the action.

You may want to put some POKEs in your HELLO file on some disks to install the SHOW command. If so, this is what they might look like:

```
100 DATA 21,42319,32,163,162,169,141,32,240,253,32,142,
         174,240,5,32,140,166,208,243,76,252,162
110 DATA 23,44686,173,0,192,16,17,141,16,192,201,141,
         240,10,173,0,192,16,251,141,16,192,201,141,96
120 DATA 4,43140,83,72,79,215
130 DATA 2,43273,32,48
140 DATA 0
150 READ N : IF N THEN READ A : FOR I = 1 TO N : READ D
    : POKE A+I-1,D : NEXT : GO TO 150
```

I tried several other versions, with features like clearing the screen, filling it up, and waiting; a stand-alone program, rather than a DOS command; and so on. You will probably want to try your own experiments.

Hierographic Transport (review).....Mike Laumer

Hierographic Tranport is a Hi-Res printer dump program for the Epson series of printers (MX-70, MX-80 and MX-100). The program is a very easy to use, menu driven system. The user manual is only 12 pages long, but most functions are self-apparent. I used the program for over an hour before I felt the need to refer to the manual. The program allows very complete control over the dot graphics mode of the Epson printers.

From the menus you can load a Hi-Res picture into either page 1 or page 2. Selections are provided for normal/inverse picture, normal/rotated pictures, normal/compressed print mode and a setable left margin to allow centering a picture on the page.

You can control magnifying or scaling the picture from 1 to 99 times normal size in the X or Y directions. This magnification is performed by repeatedly printing each screen dot, in the X and Y directions. The magnification only affects the printed image and not the screen image.

There is also the ability to select a "window" from the Hi-Res Screen that will be printed on the printer. That way you can print the rectangular section of the screen that you are interested in.

The "window" is controlled with two sets of cursor control keys. The "WASZ" keys control the top and left sides of the cursor. While the familiar "IJKM" keys control the right and bottom sides of the cursor. This is adequate for controlling the "window" but I would have prefered one set to control inward movement of the cursor sides and the other set to control outward movement of the cursor sides.

The cursor is presented as a set of blinking lines overlayed on the picture image. This technique uses the HXPLOT function described in the June issue of AAL. This function allows non-destructive lines to be drawn and erased over the top of an image on the Hi-Res screen.

The cursor lines are automatically stepped by an amount from 1-9, selectable by the number keys. The space bar or any other valid command key will stop the cursor from advancing. If "0" is selected for the step distance, the cursor lines will step by 1 whenever a cursor control key is pressed. This allows a fine positioning mechanism.

Once a "window" is selected the user can have it printed on his printer. When this is selected, the program automatically checks up on the parameters you have selected and computes the size of the image as it should be on the printer. If you have scaled the image too big, an error message will result.

The overall design of the program is good. There are however, a few minor problems in operation of the program. When the "window" is very large the automatic steps in advancing the "window" occur slowly. As the size of the "window" gets

DOS FILE EXCHANGE (DFX) $45.00

This has to be the neatest APPLE-to-APPLE DOS 3.3 file transfer program that you have ever seen! For the low price of $45.00 we will supply you with a DFX disk that contains a special utility that can transmit and save-to-disk a secondary copy of the DFX to a remote APPLE II. Thereafter, you may both dial-out, auto-answer, check-off any standard DOS 3.3 files for error checked transfer and then see both data streams or HIRES pictures as they arrive. All this and you may both exchange messages at the same time! There is no need to sit there doing nothing while a long selection of files is being transferred, type away, it will not interfere with any of the file data. Documentation is supplied as a print program and a DOS TEXT file so that you and your remote friend may reproduce your own hard copies.

REFLEXIVE VC (RVC) $45.00

You can synchronize two VisiCalc programs over the phone! We send you two RVC disks so that you and the person at a remote site may place a call between your APPLES, run the RVC program and then you both load your 16 sector VisiCalc program disks. Now the magic begins! If you both insert disks that contain exact copies of the same VisiCalc data files, either of you may then load a file and start manipulating the VisiCalc screen and BOTH APPLES will react in exactly the same way at the same time! You can even type messages to each other. Also, should either one of you initiate a VisiCalc save-to-disk command, both of you will have disk copies of the current modified VisiCalc model. RVC is virtually transparent to VisiCalc commands so now you both have a very powerful TELECONFERENCING system. Each RVC disk contains a program that prints the documentation and an RVC demonstration file.


DFX and RVC both require 48k APPLE II+ or APPLE II with Language card or APPLESOFT in ROM, a Hayes Micromodem II in slot 2, and Disk II in slot 6. You will also need to have a copy of the 16 sector VisiCalc program disk from VisiCorp at each end to run RVC.

RVC and DFX make an unbeatable combination so if you order them both at the same time, we will be happy to pay shipping charges. Otherwise please add $2.00.

To order, please write or phone:-

ARROW MICRO SOFTWARE
Box 13252, KANATA, ONT. CANADA K2K 1X4

(613) 592-4609

* VisiCalc is a registered trademark of VisiCorp.
* APPLE II is a registered trademark of Apple Computer, Inc.
* HAYES MICROMODEM II is a trademark of
Hayes Microcomputer Products, Inc.

smaller, the speed of the automatic advance gets very fast making it hard to stop on the exact point you want. The cursor routine needs a delay which varies by the size of the "window" to help even out the speed of the automatic cursor advance.

There is a record of data kept at the bottom of the screen when you are selecting a "window". This data provides you with the cursor locations and a unique display of the computed size of the picture to be printed. As the cursor is moved, the data is updated to the new recomputed picture size. The size display often flickers because blanks are written to the screen and then the data is written. If the data were written then the line cleared to the end of line, the flicker would be less noticeable.

The size display had the only bug in the whole program that I could find. The bug is rather trivial and does not affect the quality of the program. A bug, however, is a bug! [ I am sure they will fix it, once they read this review. ] When a very large scale factor (99 x 99) is used, the routine to print out the size goes bananas and displays some garbage characters on the screen. When compressed printing is selected (where the dot spacing on the Epson goes from 1/60 of an inch to 1/120 of an inch on the horizontal direction), the size display goes one character too far and scrolls the data up the screen. As the cursor window is moved arround the scrolling eventually scrolls the title lines off the main menu.

Unless you plan to print a wall mural for the side of your barn, you should never encounter the problem. A 99 x 99 scaling factor will give a pixel size of 1.5 inches square! A full screen print would be 38 feet by 21 feet in size!!! That's way beyond the carriage width of even the MX-100. The program could handle it though as long as you print it in narrow window strips. (A nice future enhancement would be for the program to automatically print an oversize picture in strips sized for your particular printer.)

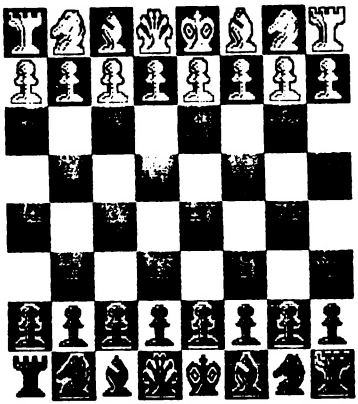
The program has a built in configuration routine and can easily be configured for the following interfaces:

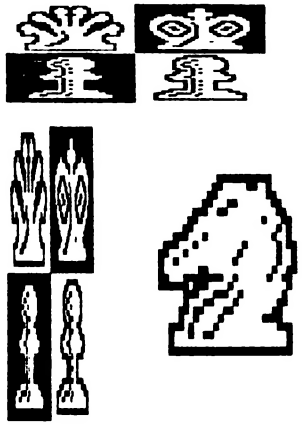
Epson APL
CPS Multi-function
Grappler
Micro Buffer II
Prometheus
Apple parallel
Epson APL (modified for 8 bit Transmission)

The Epson printers are very popular, but many more brands of printers are now on the market which have comparable capabilities. For example, the NEC PC-8023, the MPI-88G, and the Okidata series. I hope that the GSR folks come out with equivalent "Transports" for these other printers. All of them on the same disk would be especially nice!

Conclusion: A fine program for graphics printer dumping. I rate this program a "B+". A little attention to its few problems would raise the grade to "A".

This program is sold for $39.00 and is available from GSR Associates, P.O. Box 401462, Garland, Texas 75040. (Don't be afraid of the P. O. Box, they are real people.)





Christmas in July?.....Bob Sander-Cederlof

Mike Laumer has decided to offer a special price to readers of Apple Assembly Line on his FLASH! Integer BASIC Compiler. For a limited time, AAL readers can buy FLASH! for only $49, a savings of almost 40% from the normal $79 price. The offer expires September 1, 1982, and is limited to one per customer. To qualify you must mention that you read about it in AAL. and call or write directly to Laumer Research. Mike's phone is (214) 245-3927; write to 1832 School Road, Carrollton, TX 75006.

What a bargain! The FLASH! compiler is an incredible software design tool which can translate Integer BASIC programs into extremely fast machine language programs. It is the only full feature compiler on the market that can provide assembly language listings and source files compatible with my S-C Assemblers.

Synergistic Software is now selling the Galfo Integer BASIC Compiler for $149; it is copy protected, has no assembly language output, fewer extensions to the language, an undocumented run-time package, and no option to buy the run-time package source code. I have heard that it is a good compiler, but I think the price is too high.

FLASH!, on the other hand, is NOT copy protected. You can make as many copies for your own use as you need. FLASH! adds features for hi-res graphics and system programming to the Integer BASIC language. The FLASH! run-time package is fully documented, and owners of FLASH! can get the source code of the run-time package on disk for only $39. FLASH! allows easy relocation of the object code for any requirements. Used in combination with the S-C Assembler. you can further optimize the object code for even greater memory and time savings. And at this special price, it truly is a bargain. Christmas in July!

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P. O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is $15 per year in the USA, sent Bulk Mail; $18 per year sent First Class Mail in USA, Canada, and Mexico; $28 per year sent Air Mail to other countries. Back issues are available for $1.50 each (other countries add $1 per back issue for postage). 